

8. Validating Underlying Knowledge

From “Verification, Validation, and Evaluation of Expert Systems, Volume I”

If there are errors in the knowledge from which a knowledge base is built, there will usually be errors in the performance of the expert system. This chapter discusses methods for validating the knowledge from which a knowledge base is built.

Introduction

If there are errors in the knowledge from which a knowledge base is built, there will usually be errors in the performance of the expert system. There are several ways that the KB can come to represent incorrect knowledge:

- The expert(s) provide incomplete or incorrect knowledge.
- The knowledge engineer fails to correctly understand or code the expert's knowledge.
- Formalizations of knowledge, e.g. using the range of a variable to test for some underlying condition, may fail to capture all instances of the underlying condition.

There are two kinds of validation that must occur on a knowledge base: *logical* and *semantic*. Logical validation checks how the rules and objects work together to reach logical conclusions. In particular, logical validation checks for *consistency*, i.e., that all the conclusions of the knowledge base can be true at the same time. Logical validation also checks for *completeness*, i.e., that the knowledge base reaches a conclusion for all inputs. While earlier chapters of the Handbook focused on logical completeness and consistency, this chapter addresses semantic correctness and completeness.

Logical completeness and consistency are necessary for a knowledge base to be valid. However, logical completeness and consistency are not sufficient for knowledge base validity. For example, Knowledge Base 1 in the Introduction is logically complete and consistent; it contains no logical errors. However, KB 1 makes investment decisions based only on risk tolerance and discretionary income. It uses no information about actual income, debt, fixed expenses, age and other important inputs to good investment decisions. In other words, while KB 1 is logically correct, it is seriously semantically incomplete. To be valid, a knowledge base must be *semantically complete*, i.e., it must base its decisions on all information considered to be relevant by the expert.

An exception is that thorough testing (see Chapter 10, “Testing”) may show that some information can be left out without affecting performance. However, knowledge that the expert thinks is needed should be included until testing shows that an expert system performs correctly without that knowledge.

Similarly, a knowledge base can be logically consistent but not semantically consistent for its intended application. *Semantic consistency* occurs when all facts, rules, and conclusions of the knowledge base are true for the application for which the expert system is intended. To illustrate the difference between logical and semantic consistency, consider ordinary Euclidean and spherical geometry. Both are

logically consistent mathematical systems from which logically consistent expert systems can be built. However, for everyday life, Euclidean geometry is consistent and spherical geometry is inconsistent with observed facts, while for long distance navigation, the reverse is true.

It is important to note, however, that a knowledge base that is logically inconsistent by definition gives contradictory advice and is therefore semantically incorrect. Likewise, a knowledge base that is logically incomplete fails to provide a solution under some circumstances, and is semantically incomplete. **Logical completeness and consistency are prerequisites for semantic validation of a knowledge base.**

Validating Knowledge Models

Knowledge models are used as a major component of correctness proofs, but these proofs are worthless if the underlying knowledge about the application domain is false in the domain. Therefore, it is important to validate the knowledge models with domain experts.

There are several ways to validate a knowledge model:

- Use a knowledge model from a standards document in the domain. The standards process that created the model can be assumed to have validated the knowledge in the standard.
- Create a knowledge model through the joint development and consensus of a team of recognized experts in the domain. For example, the knowledge base for Quick Medical Reference, an internal medicine advisor, was created by a series of specialized consensus committees in very specialized fields (e.g., Hepatitis B). The knowledge model created in this way can be assumed to contain the best available expertise, and the participation of multiple experts increases the chances that one of them will catch any error that creeps into their discussions.
- Create a knowledge model with a single expert and review the knowledge model with other experts.

When creating a knowledge model using a single expert where correct performance is critical, it is important to validate this knowledge with outside experts not connected with development of the model. The following steps detail the validation process of the knowledge model:

- Present the knowledge model to the outside experts. In some situations it may be advisable to have someone other than the domain expert, author of the knowledge model, do the presentation, to ensure that professional courtesy does not interfere with critiquing the knowledge model.
- Collect all questions, comments, and objections to the knowledge models, or parts thereof.
- Sort and organize these comments into questions about parts of the knowledge model.
- Organize the questions into a cultural consensus test (see the following sections) to validate individual items.
- Give the test to the outside expert, to determine the extent of agreement on each of the items.
- If some of the items are not validated, perform additional knowledge acquisition and modification of the model to resolve the problems pointed to by the invalidated items. This may include additional discussions, bringing in more experts, literature searches, or redoing parts of the model.

Note that in validating the knowledge model, or in other knowledge validation activities, **it is important to ensure that the specialized expertise of experts used in validation cover the intended domain of the expert system.** Most technical fields today are too big and complex to be mastered in their entirety by a single expert, or even a few experts. Therefore, in critical applications, it is important to validate every part of the knowledge base with experts in that particular specialty. An example of this careful validation was the construction of the Quick Medical Reference expert system for internal medicine, and its predecessor systems Internist and Cadeusius. Although the final system contained nearly a thousand diseases, groups of specialists in particular diseases (e.g. hepatitis B) were brought in to collectively discuss and validate the knowledge base in their particular area of special expertise.

After performing these validation steps it is important to assess the performance of the domain expert (see the later section, Overall Agreement Among Experts). If the current domain expert differs from a consensus of other domain experts, then there are two possible courses of action:

- Replace the domain expert with one who represents a consensus of current domain knowledge.
- Continue the expert system with the disputed knowledge model, with the realization that the system will not reflect a consensus of expert knowledge. In this case it is unlikely that the system will perform in a way that matches a consensus of domain experts. Continuing development is a legitimate course in experimental or non-critical systems but is not advisable in critical expert systems.

An expert system containing knowledge which has not been validated should be used only for applications where there is no serious consequence of an error by the expert system.

Validating the Semantic Consistency of Underlying Knowledge Items

Even if the expert knowledge has been properly encoded into an expert system knowledge base, the KB will probably produce errors if the underlying expert knowledge is wrong. Therefore, it is important to validate the expert knowledge behind the knowledge base. This is particularly important because there are a number of ways in which errors can creep into the knowledge on which an expert system is built. Some of these errors are:

- The expert is wrong or out of date; in fact, all experts are probably wrong or out of date on a few points.
- The knowledge base was correct when written, but knowledge has changed.
- The knowledge engineer misunderstood the expert.
- Errors were introduced in maintenance.

When a given a fact that has been encoded into the knowledge base, how can one validate that this represents correct expertise? One approach is to do an experiment so that:

- One outcome is expected if the fact represents currently accepted expertise.
- Another outcome is expected if the fact does not represent currently accepted expertise.
- There is a statistical test that discriminates to an acceptable level of confidence between these two cases.

The specialty of cultural consensus within anthropology provides techniques for validating knowledge in a statistically rigorous manner. These techniques can be applied to knowledge validation for knowledge bases as explained below.

The basic method for validating a knowledge item is:

- Ask a panel of experts whether it is true or false.
- Tally the TRUE/FALSE answers.
- Analyze the results statistically.

Creating a TRUE/FALSE Test

In asking the experts to decide if the knowledge item is true or false, it is important not to bias them by letting the expert know which answer agrees with the current assumption in the knowledge base. Do not, for example say, "You agree with this, don't you?". To present the items for validation in a context in which both TRUE and FALSE are a priori equally likely, disregarding the truth of the item(s) being tested, do the following:

1. Start with a collection of TRUE/FALSE questions about half of which are true and half of which are false, and which are about the domain of the knowledge base. It is important that these environment-creating questions are indistinguishable by the test taker from the questions that actually test KB knowledge.
2. Scatter TRUE/FALSE questions that actually test KB items throughout the list of environment-creating questions.
3. Adjust the test if necessary so that TRUE and FALSE have approximately equal probabilities of being right.

Although this method is adequate for the purposes of this handbook, more detailed information about constructing unbiased tests can be found in literature about survey and test design.

Giving the Test

In applying the cultural consensus method to knowledge base validation, there are some issues that must be handled carefully to get maximum information from the test. First of all, the knowledge engineer must realize and explain to the experts that it is not they but the knowledge base that is being tested. The items on the test represent assertions on which the knowledge base is based, and these are being validated by experts. The reason for using multiple experts is not a lack of confidence in any one expert, but a desire to validate assumptions made in the knowledge base to a statistically significant confidence level. It is important to explain this to all the experts used in knowledge base validation to ensure that no hostility toward the knowledge engineer or the project develops. Such hostility that would rob the project of valuable contributions to the knowledge base by the expert.

Secondly, the experts used for validation should be carefully instructed to call an item false if it is not always true. This is to guard against the very real possibility that some of the rules in the knowledge base have entry conditions that are too broad. The test can even be given in a form where there are three answers to each question, TRUE, FALSE and SOMETIMES TRUE. SOMETIMES TRUE and FALSE can be combined as FALSE, i.e., the item was not considered true, when the test is scored.

Formulating the Experiment

Once the test for the knowledge base items has been written, an experiment must be constructed using the test results to validate the items. To do this, the test must be given to a group of experts to evaluate and score the results.

The test must be given to enough experts so that the correctness of each knowledge item based on test results can be distinguished from chance test results. Following is a simple statistical method to validate knowledge base items.

Analyzing the Test Results

A knowledge base item is statistically validated if:

- A majority of the experts answer that the KB item is true (or otherwise supply the test answer(s) that one would predict under the assumption that the experts think the KB item is true).
- The majority is so overwhelming that if the experts did not think the KB item was true, the chance of having results that at least this strongly suggest a belief in the KB item is less than some preassigned threshold, traditionally 5 percent or 1 percent.

Table 8.1 shows the chance of finding unanimous agreement given the "null hypothesis," that the experimental results are due to chance rather than belief in the KB item.

Table 8.1: Confidence Level

NUMBER OF EXPERTS	CONFIDENCE LEVEL
1	50%
2	75%

3	87.5%
4	94.75%
5	96.88%
6	98.48%
7	99.22%
N	$1 - 1 / 2^{**}N$

This means that it is probably a good idea to ask at least four experts to verify each important assumption backing up the knowledge base. When four or more experts agree unanimously, the assumption is reasonably validated. Six to seven experts agreeing provides a high level of confidence in the assumption.

Table 8.2 shows the confidence levels results when one expert disagreeing with the rest of the group:

Table 8.2: Confidence Levels with One Expert Disagreeing

NUMBER OF EXPERTS	CONFIDENCE LEVEL
1	0%
2	25%
3	50%
4	68.75%
5	81.25%
6	89.06%
7	93.75%
8	96.48%
9	98.05%
10	98.93%
11	99.41%
12	99.68%

This means that when one expert out of eight disagrees the KB item is validated to a reasonable level and is validated to a high level when one expert out of ten disagrees.

In general, if there are N experts of which M disagree, the confidence level achieved by this level agreement is:

$$1 - (1 / 2^{**}N) * \text{SUM}(m = 0 \text{ to } M) \text{combinations}(M, N)$$

where combinations(M,N) is the number of combinations of M objects chosen from N.

This is computed by:

$$\text{combinations}(M, N) = M! * (N - M)! / N!$$

where $K!$ is the factorial of K .

Overall Agreement Among Experts

The above method of validation based on cultural consensus rests on an assumption that the experts share the same basic knowledge, i.e., the same ideas about how to solve the problems covered in the knowledge base, and are validating the specifics of that common approach, as expressed in the knowledge base. Sometimes, however, experts do not agree in their basic knowledge and approach to a class of problems. To detect whether all the experts take the same basic approach to problem solving, observe the following:

1. **Cluster the experts:** Represent each expert as the vector of answers on the TRUE/FALSE test. Find a clustering of the experts based on these vectors.
2. **Test for similarity:** Test to see if all the experts belong to the same cluster.
 - 2a. **Common cluster:** If all the experts belong to the same cluster, then the computation of item confidence presented above remains valid.
 - 2b. **More than one cluster:** If there is more than one cluster among the experts, analysis of the differences among experts must be conducted, as discussed below. Then the cultural consistency of individual KB items should be retested.

For the small number of experts that are involved in validating a knowledge base, clusters of experts can be determined by hand inspection of the correlation matrix of test answer similarity of experts.

Approaches to Disagreement Among Experts

When experts do not agree, as evidenced by the existence of more than one cluster of experts, the following approaches are useful:

1. **Throw away outliers:** If it can be determined by interviewing other experts that an expert who is not part of a larger cluster of experts represents a little-held school of thought within their specialty, and if the more mainstream approach represented by the large cluster of experts successfully solves the problems for which the expert system is intended, eliminate the outlier expert from the validation sample of experts.
2. **Choose a valid subset of experts:** If two clusters of experts work from totally different assumptions, pick a cluster that achieves optimal results and use them both as the source of domain expertise and experts for validation. Do not try to include two conflicting schools of expertise in the same knowledge base.
3. **Use the separate approaches as subsystems:** If approaches represented by distinct clusters of experts do better on different subsets of the target domain, it may be possible to build a system where the differing approaches reside in separate expert subsystems. These subsystems could participate in a weighted vote to determine an overall conclusion, where the weight given to a vote is the heuristically determined confidence factor that a particular subsystem can solve the problem

under consideration. Since this approach leads to a more complex, expensive system, it should only be used when the separate approaches are not adequate by themselves.

4. **Analyze disagreements:** Two or more clusters of experts may be a symptom of unresolved controversies within the professional specialty supplying the expertise for the expert system. In this case, the expert system development team needs to decide if there is enough agreement among experts to build an expert system that gives reliable advice in the domain for which it is intended.

Clues of Incompleteness

Clues that a knowledge base is semantically incomplete may exist within the knowledge base itself. One is that the knowledge base is logically incomplete. Another is that variables, statements, conclusions, etc., are defined but not used. This may indicate that an expert started to supply knowledge that would use them, but never completed that part of the knowledge base. Therefore, the entire knowledge base should be checked for items that are defined but not used, and each one of these should be used or eliminated on expert advice.

Variable Completeness

Variable completeness is a special case of semantic completeness. A knowledge base is variable complete if it uses all of the important input variables in making its conclusions. The steps in checking variable completeness are:

1. Determine and codify what inputs the KB uses in determining each variable and the truth of conclusions.
2. Ask experts to confirm the knowledge codified in Step 1.

There are two ways to determine and codify the variables used in making decisions:

- Computerized analysis of the knowledge base.
- Keeping careful knowledge acquisition and coding notes.

In either case, the goal is to be able to formulate questions of the form:

- The knowledge base currently uses variables V1,V2,V3...VN to decide X.
 - Are there additional variables that should be in this list of inputs? What are they?
 - Are there input variables that are not needed? If so, then what are they?

Once these questions have been defined they should be presented to experts, possibly first to the experts used in building the expert system, and then to independent experts. The process of asking experts about input completeness should be continued until the variable set stabilizes. Then the variable sets should be validated using the technique described above for knowledge item validation.

Semantic Rule Completeness and Consistency

Once the inputs to making decisions have been validated, the actual rules that make each decision should be validated. One problem in validating knowledge bases has been that the size of knowledge bases and their relative lack of easily perceived structure makes them difficult for domain experts to read. To lessen this problem, the knowledge base can be partitioned into the pieces that determine the value of each important variable and conclusion. Each such piece represents the knowledge in the knowledge base about a particular subtopic of the domain, and some conclusion drawn from that subtopic. The expert(s) is asked to examine each piece of the knowledge base separately, and answer the following questions:

1. Is the information expressed in the rules that set the value of some particular variable or statement correct?
2. Is the information complete? Or are there other conditions to consider, either in individual rules or as new rules?

By focusing the expert's attention on a single variable at a time and the conditions for setting that variable, a large knowledge base is broken down into pieces that are easier to comprehend.

A backwards chaining strategy can be used to go through the variables and statements in an order that is logical to an expert. Start with the overall outcomes of the knowledge base, and for each pull out all the rules that set that conclusion. Validate these rules. Then do the same for rules that set the conditions in the "if" parts of validated rules. Continue the backward chaining validation process until validated pieces cover the entire knowledge base. The question, "Is this knowledge base piece valid", i.e., is the information correct and complete, can be considered a knowledge item, and validated to the desired level of confidence using cultural consensus, as discussed above. For knowledge bases where reliability is critical, this piecewise validation should be carried out.

Validating Important Rules

Particular emphasis should be placed on validating rules that cover and appear to cover many inputs or which process critical cases. Rules that appear to cover many cases are those with few atomic formulas in their "if" parts. These rules should be pulled out and validated by experts.

To determine which rules typically handle common cases the knowledge engineer in charge of validation should collect a set of typical input data from one or more experts. Each data set is run on the expert system, keeping track of which rules fired in processing this data. Those rules are presented to the experts for validation.

Exactly the same process is used to validate critical cases; data sets are gathered from experts, the data sets run, and the firing rules validated by the experts.

Validating Confidence Factors

Rule bases may contain assertions about the confidence of conclusions under various conditions, as illustrated by this rule from PAMEX:

```
if DS = 14
    and NOT Deterioration Cause Indicator = Structural Failure
    and NOT Deterioration Cause Indicator = Weather Severity
    and Skid Number = Low
    and DV2 >= 15
    and DV15 < 30
then conclude Aggregate Spray, confidence = 0.8
and conclude Open Friction Course, confidence = 0.8.
```

A problem in validating the knowledge base is to insure that the confidence values are semantically consistent. In particular, if three rules with many "if" conditions in common have confidence values for a conclusion of, for example, 0.9, 0.85 and 0.5, it is important to insure that the low confidence factor is justified by domain knowledge. Either through a coding error, or because different experts supplied the confidence factors, it is possible that the large difference is an artifact of building the expert system.

The basic strategy for validating the confidence factors is:

- Predict the confidence factors for rule conclusions by estimating them heuristically from the conclusion confidences of similar rules.
- Compare the predicted confidences to those actually written into the knowledge base.
- Validate the confidences where the predicted and actual differ by more than some threshold.

The first step in implementing this validation consists of rule simplification. The following rule simplifications should be carried out before predicting confidence factors:

- From a rule of the form "if A then B and C", form two rules, "if A then B" and "if A then C", so that the confidence factors of B and C will be validated separately.
- Normalize the relational operators by:
 - Replacing all < and <= operators with > and >= operators.
 - Replacing $X \geq Y$ with $X > Y$ OR $X = Y$.
 - Replacing $X \neq Y$ with NOT $X = Y$.
- Write the "if" parts of rules in disjunctive normal form, i.e., as an OR of ANDs of atomic formulas and negations of atomic formulas.
- From a rule of the form "if A OR B then C" form two rules, "if A then C" and "if B then C", so that the two conditions A and B can be validated separately.

The predicted confidence factors are based only on rules having the same conclusion, i.e., to validate the confidence factor of B in "if A then B", it is only necessary to look at other rules with conclusion B. Therefore, although the rule simplifications multiply the number of rules, partitioning by conclusion breaks the rules into subsets of manageable size.

Confidence factors are assigned to atomic formulas in rules in a two-step process. The first step is to assign confidence factors to the atomic formula itself. The second step is to modify that confidence factor if the atomic formula is the argument of a NOT. If an explicit confidence factor appears with an atomic formula, use that as the initial confidence factor for the formula in a rule. Otherwise, if an atomic formula appears in a rule, use 1 as the initial confidence factor. If an atomic formula does not appear in a rule, use 0.5 as its confidence factor. Now, having defined confidence factors for the atomic formulas themselves, modify them to account for NOT's as follows: if an atomic formula with initial confidence C is an argument of NOT, its confidence is 1-C; otherwise, its confidence is C.

At this point, a confidence factor has been assigned to every atomic formula in every rule "if" part. Given two rules, R1 and R2 for each atomic formula A, let A1, A2 denote the respective confidence factors. Then define:

$$\text{distance}(R1, R2) = \text{sqrt}(\text{SUM}(\text{atomic formulas } A)(A1-A2)^2))$$

i.e., the square root of the sum of squares of difference between corresponding confidence factors. Using this distance, an estimated confidence factor can be conducted by using a generalized regression neural network (GRRN), which is described in the appendix to this chapter.

In interpreting the differences between actual and estimated confidence factors, it must be decided how much difference should trigger validation. Small differences of 0.1 and possibly 0.2 probably represent expert judgments. Larger differences may indicate errors in the knowledge base, but may also indicate valid expertise. Confidence factors with large differences between predicted and actual values should be validated in a two-step process. First validate the confidence factors with a single expert, e.g., the project domain expert; secondly, if doubt remains, validate the confidence factors with multiple experts using cultural consensus. Since differences may represent expert knowledge, if the expert validates a confidence factor, it may be accepted as valid, or at least as valid as any other knowledge item supplied by the expert. Like other knowledge items, the single-expert-validated confidence factors may be further validated by multiple experts. However, most of these confidence factor differences reflect the fine structure rather than the major assumptions of knowledge bases, and the priority of validating most of them is small. If the difference is large and the consequence of the difference is judged to be serious, however, the confidence factor should be validated by multiple outside experts.

Given that resources are always limited, it is impossible in practice to validate all the items in a knowledge base. Given the need to triage testing, it is important to note during knowledge acquisition:

- which areas of the knowledge base are the most controversial among experts
- which experts disagree most with their colleagues.

In addition, to select priority items for testing, it is important to perform a hazard analysis of the system containing the expert system. This analysis should extend into the expert system, and define which assumptions in the knowledge base are safety critical.

Given both general areas of disagreement in the knowledge base, and priority areas for safety, the knowledge engineer can set priorities for testing underlying assumptions. It is very important to test

items that are both safety critical and prone to expert disagreement; a system that reasons correctly from false information is likely to fail.